Semi-symbolic Simulation and Analysis of Deviation Propagation of Feature Coordination in Cyber-physical Systems

Michael Rathmair, Christoph Luckeneder, Hermann Kaindl
TU Wien, Vienna, Austria
{rathmair, luckeneder, kaindl}@ict.tuwien.ac.at

Carna Radojicic
TU Kaiserslautern, Kaiserslautern, Germany
radojicic@informatik.uni-kl.de

Abstract

For studying the effects of deviations for uncertain inputs of systems, often multi-run simulation is employed, which is time-consuming. Unfortunately, such simulations also do not directly support the traceability of such effects. A semi-symbolic modeling approach based on Affine Arithmetic Forms *allows the representation of uncertainty in terms of ranges. Simulations of such models directly include propagation of deviations and their traceability. This paper presents such a semi-symbolic model of a cyber-physical system including coordination of safety-critical and interacting features. For feature coordination, this model introduces handling discrete uncertainty with two different behavioral modes and their integration. Based on this model, a single simulation run allowed us studying the effects of several deviations. In addition, this modeling approach facilitates specific analyses of deviations based on the traceability information. As a result from simulation and analyses, we got a better understanding of the different deviation propagations within our model.*

1 Introduction

In a previous simulation study of a cyber-physical system [6], we provided new insights into *feature coordination*, which is more intricate in such a context than, e.g., in telecommunications as studied already long time ago in [5]. However, this study only involved single simulations runs with presumably exact values of physical variables.

In reality, there are usually deviations, which would usually be dealt with through multi-run simulations. These would be time-consuming, however, and they would not directly facilitate specific analyses of deviation propagation.

Hence, we employed a different modeling and simulation approach based on *Affine Arithmetic* (AA) [10]. AA is an extension of classical Interval Arithmetic and may be used in any field of technical calculations or simulations. Our motivation for using AA for modeling feature coordination in cyber-physical systems has been twofold:

- We have been interested in studying deviations and their propagation in such models, and
- we wanted to extend the scope of this modeling and simulation approach based on AA to systems with discrete uncertainty.

Discrete uncertainty arises in our models through calculations of the minimum of values that represent uncertainty. Hence, we had to provide a minimum operator for this modeling and simulation approach. Actually, we defined and implemented two different approximations of a minimum operator and also studied their different effects.

As a running example, we use *Adaptive Cruise Control* (ACC). This is an automotive driver-assistance feature implemented in modern upper-class vehicles. ACC has, in principle, two functional modes. First, it may act as *Cruise Control* (CC), keeping a user-selectable cruise speed of the vehicle (largely) constant. Second, it may act as *Distance Control* (DC) of a vehicle A following another vehicle B, keeping a minimum target distance (in reality, a time span), so that a potential rear-end collision is to be avoided. While ACC is already well understood as a single feature from an engineering viewpoint [11], we model it as a *composite feature* that coordinates CC and DC. Still, we follow the engineering-approved coordination scheme selecting the minimum speed request of both features [11], which was also shown to work as desired in [6, 8].

Much as in [6], we study here only the typical scenario where Vehicle A is approaching first until the selected target distance is reached, while vehicle B does not change

URI: http://hdl.handle.net/10125/50597
ISBN: 978-0-9981331-1-9

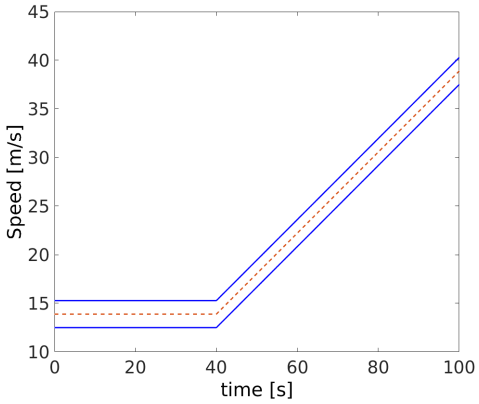

Figure 1: Speed profile of vehicle B represented as an Affine Signal

its speed according to the left part of Figure 1. Vehicle A keeps the distance until vehicle B accelerates (according to the right part of the figure). Vehicle A may also increase its speed now, but it must not exceed the selected CC cruise speed.

For studying all possible behaviors and the corresponding coordination of features, *model checking* (see, e.g., [1]) is preferable to simulation. However, due the combinatorics involved, only a minimalist qualitative model of the cyber-physical system was used for this purpose in [8].

The remainder of this paper is organized in the following manner. First, we provide some background on Affine Arithmetic, in order to keep our paper self-contained. Then we present our ACC model using Affine Arithmetic Forms. Since it requires a minimum operator not yet available in the simulation environment, we also propose two different approximations for providing such an operator. Based on all that, we present results from our simulation and related analyses.

2 Background on Semi-symbolic Simulation using Affine Arithmetic Forms

Affine Arithmetic was originally developed for computer graphics [2] and is an extension to classical Interval Arithmetic. It is applicable to any field of technical calculations and simulations where parameter deviations representing uncertainty occur [10]. An *Affine Arithmetic Form* (AAF) models such a deviation by using an abstract *deviation symbol* ϵ_i. These symbols make the simulation approach *semi-symbolic* and not just numeric. The deviation symbols propagate through the system and are still present in the outputs of the system model [3].

An AAF is defined as a scalar real center value x_0, which is extended by a sum of linear deviations. A deviation is represented by a product of a *partial deviation value* x_i and a deviation symbol ϵ_i. Each symbol has an unpredictable value between -1 and $+1$ [3, 10]. The maximum spanned deviation interval is the *radius* of an AAF. Deviations and thus also the radius of an AAF are always located symmetrically around the central value. The set $\mathcal{N}_{\hat{x}}$ contains all deviation indices i of an AAF with $x_i \neq 0$. With this information, an AAF can be mathematically written as given in Equ. 1, and a set $\mathbb{A}$ of all valid AAFs can be defined.

$$\hat{x} = x_0 + \sum_{i \in \mathcal{N}_{\hat{x}}} x_i \epsilon_i \tag{1}$$

$$\text{with } \hat{x} \in \mathbb{A},\ x_0 \in \mathbb{R},\ i \in \mathcal{N}_{\hat{x}}$$
$$x_i \in \mathbb{R},\ \epsilon_i \in [-1, 1]$$
$$\mathcal{N}_{\hat{x}} = \{i \in \mathbb{N}^+ | x_i \neq 0\}$$

Each ϵ symbol included in an AAF represents a specific modeler-defined uncertainty cause. In this sense, a deviation at the input of the system model is marked by an abstract symbol. This offers the possibility for explicitly tracing a specific uncertainty from its cause to its consequence.

Based on the definition of an AAF given in Equ. 1, we also define an *Affine Signal*, which has a dependency in time

$$\hat{x}(t) = x_0(t) + \sum_{i \in \mathcal{N}_{\hat{x}}} x_i(t) \cdot \epsilon_i. \tag{2}$$

The central value and partial deviations representing the width of the form may be variable with respect to the simulation time.

In a standard AAF, only linear deviation is included (while a quadratic AA approach has been proposed as well [3]). Linear operations $(+, -, \cdot)$ where $\cdot$ defines a multiplication with a constant $c \in \mathbb{R}$ on AAF are [10]:

$$c \cdot (\hat{x} \pm \hat{y}) = c \cdot (x_0 \pm y_0) + \sum_{i \in \mathcal{N}_{\hat{x}}} c \cdot (x_i \pm y_i) \cdot \epsilon_i \tag{3}$$

Nonlinear functional operations on two or more AAFs may produce polynomial terms with an order higher than one, or discrete discontinuities. For an affine representation of such results, it is required that a nonlinear operation transforms the higher-order result into a valid AAF containing only linear terms. Therefore, an appropriate *approximation*, e.g., for multiplication, division, inversion, exponential function, trigonometric functions, etc. must be integrated [10].

There are several algorithms for calculating approximations for standard mathematical operations published in the literature. In principle, in an approximation form an additional deviation symbol is added to the AAF. This additional symbol extends the total range of an AAF to cover nonlinearities [3, 9, 10]. Within our framework, an approximation is represented by an α and a β value. α is the partial deviation value of the corresponding additional deviation symbol. β is the potentially shifted central value to satisfy the symmetry requirement of deviations [9].

Table 1: Parameter setup and deviation documentation of the ACC example

	Parameter	Description	Value
ACC configuration of vehicle A	d_{init}	Initial distance between vehicle A and B	$40\,m$
	v_{CC}	Cruise control speed setup	$100\,km/h\ (=27.78\,m/s)$
	$v_{CC,1}$	Deviation of the cruise control speed setup	$\pm 2.5\,km/h$, $(=\pm 0.694\,m/s)$
	d_{target}	Distance control target distance setup	$15\,m$
	k	Distance controller proportional factor	1
Speed profile of vehicle B	$v_{B,init}$	Initial speed of vehicle B	$50\,km/h$, $(=13.9\,m/s)$
	$t_{accelerate}$	Time period before vehicle B accelerates	$40\,s$
	a_B	Acceleration rate of vehicle B	$0.417\,m/s^2$
	$v_{B,2}$	Speed deviation of vehicle B	$\pm 5\,km/h$, $(=\pm 1.39\,m/s)$

The simulation framework that we implemented based on Affine Arithmetic provides functions

- for the definition and management of AAF, including linear and a set of selected nonlinear mathematical operations,
- for verifying whether resulting system outputs are within specified tolerances [4],
- for analyzing the impact of partial deviations on the output signals or internal quantities of a system [4], and
- for tracing deviations and AAF representations at simulation runtime for analyses of deviation compensation and amplifying effects.

3 Modeling ACC using AAF

We build upon our previous simulation model defined in [6] and keep its physical simplifications for creating here an ACC model using AAF. Physical quantities are reduced to speeds of vehicles and the distance between them. The unidirectional dependency is given by Newton's law $d = \int_0^t (v_B - v_A)dt + d_{init}$, where d stands for the distance between vehicles A and B, v_A and v_B for the corresponding speed values, and d_{init} for the initial distance. A specific property of the dependency between speed and distance of vehicle A is the unpredictable influence of the speed of vehicle B.

Actually, this is a cyber-physical system where the coordinator of the feature ACC composed of the features CC and DC requests a certain speed (in the physical part of the model). As given above, this request obviously influences the distance. Since the distance influences DC, which potentially influences ACC, this model includes a closed loop.

We represent this model here using the block diagram in Figure 2. CC delivers a nearly constant speed value as its output. DC computes a speed request according to the

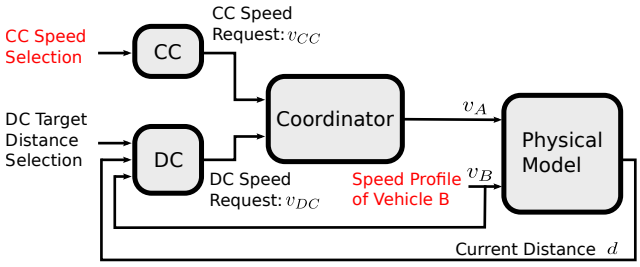

Figure 2: Block diagram of the modeled ACC system

controller formula given in Equ. 4:

$$SpeedRequest_{DC} = (d_{current} - d_{target}) \cdot k + v_B, \quad (4)$$

where k is a proportional factor multiplied by the difference of the current distance and the selected target distance. The coordinator block forwards the lower of these speed requests at each point in time (minimum coordinator behavior) v_A as the speed of vehicle A. For the driving behavior of vehicle B, we assume a static speed profile as given by Figure 1. The distance between vehicles A and B is computed as specified above. A representative part of the distance output of the model, the current distance, is fed back to the input, so that the subsequently induced uncertainty is continuously propagated over time. Apart from these linear blocks, ACC includes a discontinuity given by the coordinator behavior of switching between the system's CC and DC modes.

In contrast to [6], we performed a semi-symbolic simulation, where a specific symbol represents a defined uncertainty. We included two such symbols for parameter deviations, according to the two deviation causes shown in red in Figure 2. These are a deviating CC speed of vehicle A and an unpredictable speed profile offset of vehicle B (see the blue boundary lines in Figure 1). These parameter deviations model that in reality both these speeds fluctuate within certain interval ranges.

Table 1 contains the parameter setup of the model for our semi-symbolic simulation, including deviations and their

corresponding values. According to these values, initial AAFs $\hat{v}_A$ and $\hat{v}_B$ can be stated as $\hat{v}_A = v_{CC} + v_{CC,1} \cdot \epsilon_1 = 100 + 2.5 \cdot \epsilon_1$ and $\hat{v}_B = v_{B,init} + v_{B,2} \cdot \epsilon_2 = 50 + 5 \cdot \epsilon_2$. The uncertainty causes represented by the ϵ_1 and ϵ_2 symbols directly influence the coordinator output speed request and the calculated distance between the vehicles. Symbols are propagated through the system and fed back to the input of the Distance Control block. Thus, partial deviations of a time-step also influence future simulation time-steps. The full model is calculated with a time-step period of $10ms$.

4 Minimum Coordinator Block

For the coordination of CC and DC, we use a minimum coordinator block, which forwards their lower speed request value as applied at its inputs. In the given simulation environment, no minimum operator for AAFs has been available yet. Hence, we had to model and implement one for calculating $min(\hat{v}_{CC}, \hat{v}_{DC})$, where $\hat{v}_{CC}$ and $\hat{v}_{DC}$ are AAFs. The minimum operation models a specific discontinuity behavior, which we approximated.

Figure 3 shows the individual CC and DC speed requests when vehicle A is in CC mode and approaching vehicle B. The center value of $\hat{v}_{CC}$ is constant at $100\,km/h$, superimposed by a $2.5\,km/h$ deviation (symbol ϵ_1). The center value and the specified deviation range are illustrated by the brown dashed and the red lines, respectively. $\hat{v}_{DC}$ is decreasing according to the implemented control Equ. 4. Again, the brown dashed line indicates the center value and the red line the deviation caused by the symbol ϵ_1. Additionally, $\hat{v}_{DC}$ includes the deviation symbol ϵ_2 representing the speed deviation of vehicle B, where the blue lines show the accumulated deviation of both ϵ_1 and ϵ_2. Due to the closed-loop model structure, $\hat{v}_{CC}$ and $\hat{v}_{CC}$ are correlated through including the same symbol ϵ_1.

The rightmost simulation point in Figure 3 indicates the first point in time when the ranges of the speed requests from CC and DC overlap. In principle, starting from this point in time, due to the unpredictable values of the ϵ symbols, we cannot determine with certainty whether the CC or the DC speed request has a lower value. We call this characteristic an uncertain minimum, because we cannot determine if the output of the coordinator block derives from a CC or a DC speed request.

At a closer look, due to the given symbol correlation between $\hat{v}_{CC}$ and $\hat{v}_{DC}$, it may even happen that the point in time when the minimum of requested speeds becomes uncertain is additionally shifted in time. Hence, in a first step we handle the given ϵ_1-correlation correctly by calculating the AA subtraction of $\hat{v}_{DC}$ and $\hat{v}_{CC}$ (see Equ. 3). The result of $\hat{v_z} = \hat{v}_{DC} - \hat{v}_{CC}$ is plotted in Figure 4. The minimum becomes uncertain if the value 0 is within the calculated interval range of $\hat{v}_{DC} - \hat{v}_{CC}$. This can be expressed by using

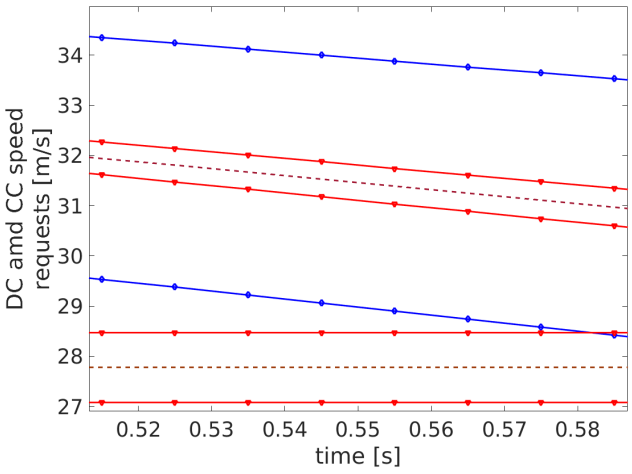

Figure 3: CC and DC speed requests and their deviations over time, until their ranges start to overlap

the functions $value$ and $range$: $0 \in range(value(\hat{v_z}))$ [7]. The arrows in the figure indicate such a situation. If and when this happens, we cannot uniquely evaluate if the ACC feature is in CC or DC mode, due to the unknown values of ϵ_1 and ϵ_2.

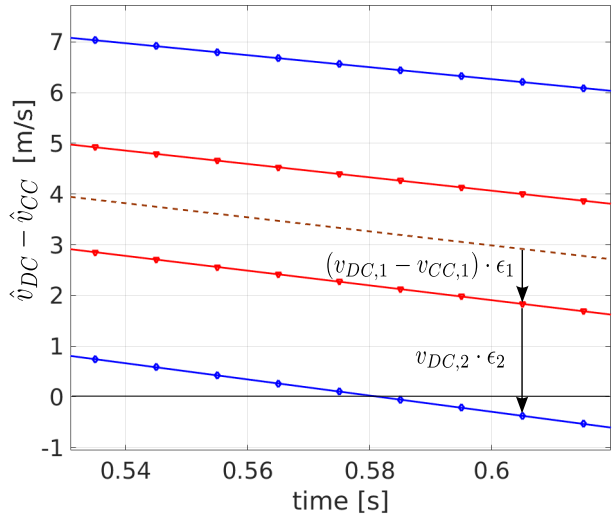

Figure 4: Plot of $\hat{v}_{DC} - \hat{v}_{CC}$ and a case where 0 is included within its interval range

4.1 Minimum Approximation with Intervals

As a second step, for points in time when the minimum is uncertain, we calculate approximations. Our first minimum approximation is based on Interval Arithmetic calculations,

which do not take correlations between CC and DC requests into account. Interval Arithmetic calculations, in general, result in more pessimistic solutions than AA calculations. Hence, we actually calculate a pessimistic-approximation range.

As illustrated in Figure 5, we have to distinguish four cases of how the ranges of CC and DC relate to each other. The ranges on the right plotted in red are the results for each case, to be used for our approximation of the minimum form $min(\hat{v}_{CC}, \hat{v}_{DC})$.

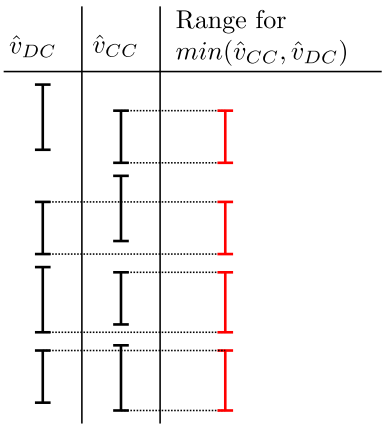

Figure 5: Cases of overlapping CC and DC ranges and the corresponding range for the minimum approximation

To create an output AAF for the minimum operator based on that, we handle the approximation representation in the same way as for nonlinear operators such as multiplication, division, etc. The affine part of the result is defined either by the CC or the DC request, depending on their central values. However, the output of the minimum operator is an AAF caused by the DC or CC request inputs, respectively. We also add an extra symbol and a partial deviation α_i, which widens this range if the minimum is uncertain, i.e., the minimum value can be taken from the other Affine input than the one that is picked for the output. For the resulting AAF, we calculate also an approximation central value β_i, which is located in the middle of the approximated output range.

As a result and to sum up, our (first) approximation for the minimum operator includes a two-step process:

- First we evaluate an AF for $\hat{v}_{DC} - \hat{v}_{CC}$. If 0 is included within its interval range, we know that the minimum value is uncertain and can be caused by a CC or a DC request, respectively. The system can be in CC or DC mode depending on the unknown value of ϵ symbols.

- As a second step, if the minimum is uncertain, we calculate an approximation form covering the worst case. The approximation interval is evaluated by uncorrelated Interval Arithmetic considerations including different cases of overlapping situations. Finally, we store the approximation within an AAF object setting according to α and β values.

4.2 Minimum Approximation with AADDs

Since the minimum approximation with intervals does not utilize any of the information contained in an AAF in addition to the intervals per se, we also developed a more precise approximation with *Affine Arithmetic Decision Diagrams* (AADDs).

As described above, the minimum is uncertain if we cannot determine if CC or DC has a lower value, i.e., if the value 0 is within the calculated interval range of $\hat{v}_{DC} - \hat{v}_{CC}$. The basic idea for handling this uncertainty is to use a binary decision diagram for minimum approximation. At each point in time during the simulation when this uncertainty exists, an internal vertex is created in such a diagram with two successor edges, labeled $true$ and $false$, depending on whether the condition $(\hat{v}_{DC} - \hat{v}_{CC} \geq 0)$ is fulfilled. The value $true$ signifies the CC case, i.e., in this part of the tree the assumption is that the speed request of CC is used. For the value $false$, however, the speed request of DC is taken. Since $\hat{v}_{DC}$ and $\hat{v}_{CC}$ are AAFs, this distinction depends on the values of ϵ_1 and ϵ_2. A solver is used for finding those partitions of epsilon ranges for which the condition is fulfilled or not. Through recursive application for the next point in time of the simulation, a tree structure is created.

For each vertex below in the tree, only these value ranges are valid, of course. For each path, there is a terminal vertex, whenever there is no minimum uncertainty any more. It is assigned the AAF of $\hat{v}_{DC}$ or $\hat{v}_{CC}$, respectively.

In fact, this approximation is analogous to the one of a "less than" operator formally defined in [7]. Hence, we were able to simply build on this definition and the implementation of this operator. However, for the minimum operator, we had to define different terminal vertices. While they are Boolean values for "less than", we have AAFs instead in our AADDs.

For a simulation using such an AADD, there is a complication involved, however. Concurrently to building this tree, the simulation actually has to simulate all the variants according to the paths of the tree. While doing so, the maximum and the minimum deviations occurring are calculated and returned as the result of this minimum approximation.

5 Semi-symbolic Simulation

In the simulation run, the speed profile of vehicle B as shown in Figure 1 is used. Vehicle A is approaching first,

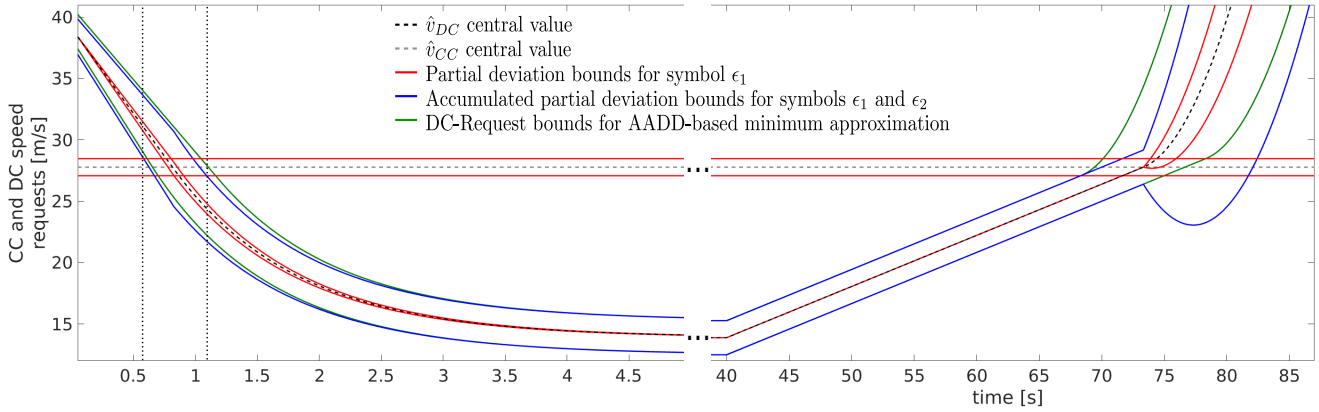

Figure 6: CC and DC speed requests

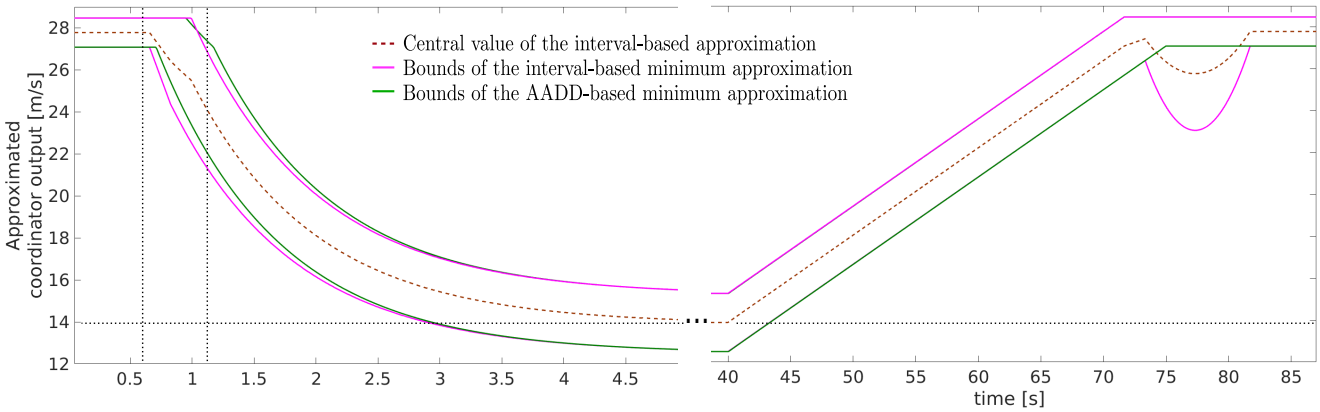

Figure 7: Approximated minimum coordinator output

while $v_A = v_{CC} > v_B$, until the selected target distance is reached. This is illustrated in the left part of Figures 6, 7 and 8. Then, as long as the speed of vehicle B does not change, $v_A = v_{DC} = v_B$, the DC block holds the given target distance. (This time period is cut out in the figures.) Once vehicle B accelerates resulting in $v_B > v_{CC}$, vehicle A may also increase its speed, but it must not exceed the selected CC cruise speed (see the right part of Figures 6, 7 and 8). Hence, there is a discrete change involved again, this time from $v_A = v_{DC}$ to $v_A = v_{CC}$.

More precisely, we ran the simulation based on AAFs. For calculating the minimum of AAFs, we used the approximations based on interval considerations and AADDs as defined above. The corresponding lines in the figures are shown in magenta and green, respectively. For showing the subintervals for the interval approximations, we use red lines for ϵ_1 and blue lines for accumulated ϵ_1 and ϵ_2 deviations.

Figure 6 illustrates the speed requests of CC and DC, respectively. CC has a constant speed request of $27.78\,m/s$ and a deviation range of $0.694\,m/s$ indicated by the red lines representing the bounds of the range. The DC feature has a decreasing speed characteristic caused by the decreasing distance between the vehicles. The blue lines show the accumulated ϵ_1 and ϵ_2 deviation causes of the DC speed request for AAF using interval-based approximation for the coordinator block. First, ACC is in CC mode, and the upper bound of the CC request is clearly smaller than the lower bound of the DC request, until the ranges start to overlap, which is marked by the two vertical cursors (strictly speaking, the $\hat{v}_{DC} - \hat{v}_{CC}$ interval range includes 0).

Starting at this point in time, the minimum operation of the coordinator block is approximated. The corresponding output of the coordinator is shown in Figure 7. As described in Section 4, we evaluate two approximations for the minimum based on intervals and AADDs, respectively. The slight differences between the green and the magenta line in the left part of the figure are caused by the behavior of the interval-based approximation, which is still switching between DC and CC modes based on the center value of the AAF.

As a result for the coordinator, we plot in Figure 7 the boundaries of the approximation forms for interval approximation (in magenta), and for AADD-based approximation

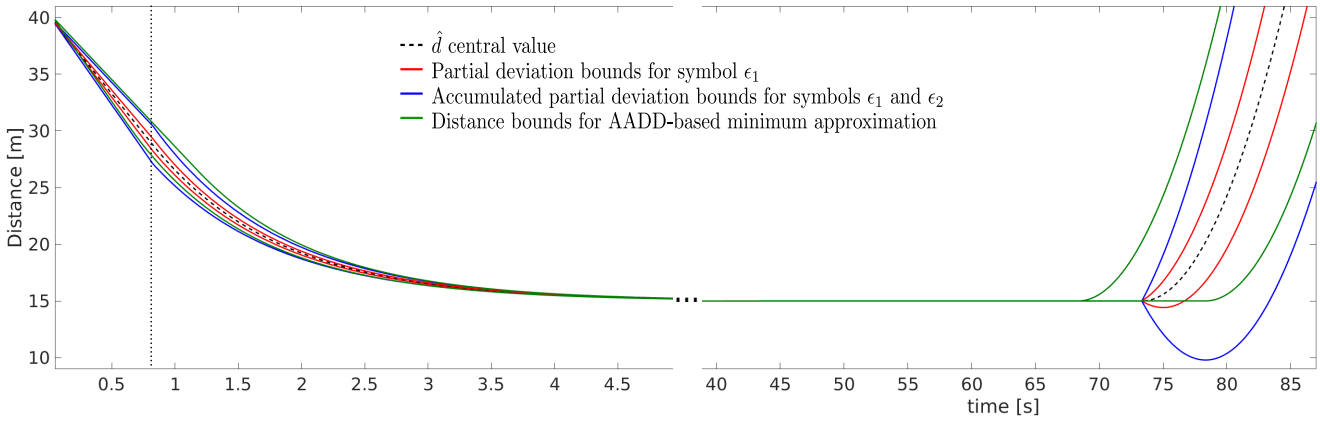

Figure 8: Distance between vehicles A and B

(in green). Due to the strict requirement of an AAF that the center value has to be in the middle of the spanned range, for the interval-based approximation a center value is additionally calculated (stored as a β value within an AAF object). It is illustrated by the brown dashed line.

Once the minimum is no longer uncertain (the DC request is for certain smaller than the CC request), ACC is in DC mode. The speed request changes its characteristic to an exponentially shaped curve, asymptotically approaching a value of $13.9\,m/s$ (see the horizontal cursor), which is the speed of vehicle B. The exponential shape is caused by the control behavior of the DC feature (see Equ. 4) and the closed-loop system structure.

The right parts of the Figures illustrate the acceleration scenario of vehicle B. We focus on the effect occurring at the point in time when vehicle B is becoming faster than the CC speed of vehicle A ($t > 68\,s$). Much as in the approach scenario, the DC and CC request ranges start to overlap and the coordinator calculates an approximation for the minimum (more precisely in two different ways). As described in Section 4, for the interval-based approximation ACC switches between CC and DC mode, depending on the minimum of the center values of their requests. Thus, ACC switches back to CC mode, but DC is still independently calculating speed requests in parallel. Due to the constant output of the coordinator (CC speed), for DC the feedback loop is broken. Thus, the deviation bounds are increasing. The coordinator is still approximating the minimum by taking the interval minimum according to the cases shown in Figure 5. This explains the partially parabolic shape of the approximation curve of the minimum values. Hence, for the described scenario where vehicle B is speeding up and exceeding the configured CC speed of vehicle A, the minimum approximation has a pronounced pessimistic range. The approximation using AADDs (green line in Figure 7) does not have this effect and is, therefore, much more precise.

Figure 8 shows the distance between vehicle A and vehicle B over time. The corresponding center value (brown dashed line) starts at the defined initial distance of $40\,m$. Since there is no initial deviation set for the distance, the first simulation point is an AAF having a radius of 0. In the following, the deviations of the CC speed (based on the symbol ϵ_1) and the speed of vehicle B (based on the symbol ϵ_2), continuously influence the range of the distance, until ACC is in DC mode and the deviation causes (partial deviation values) are constant. Due to the physical dependency between the speeds of vehicles A and B, also the partial deviation values get integrated over time. The influence of ϵ_1 is illustrated by the partial deviation bounds as shown in red boundary lines, the influence of ϵ_2 by the blue boundary lines, where more precisely the red part has to be taken out, since the blue lines show the accumulated partial deviations.

A closer look on the subinterval ranges for ϵ_1 and ϵ_2 reveals that the partial deviation with symbol ϵ_2 representing the speed deviation of vehicle B has a higher impact on the deviation range of the distance. This can be seen in Figure 8, since for all points in time (except 0) the differences between the red lines and the brown center line are smaller than the differences between the blue and the red lines. If and when ACC is in DC mode (illustrated on the right from the courser in Figure 8), the radius of the distance is decreasing. This is caused by the closed-loop system structure including the DC controller block, which continuously compensates deviations. The asymptotic trend of the distance line approaches the specified target distance of $15\,m$.

The right part of Figure 8 illustrates the distance range for the situation where vehicle B exceeds the selected CC speed of vehicle A. Based on the approximation effect for the interval-based approach described in Figure 7, there is also an impact on the distance boundaries. The parabolic shape of the curve of the approximated speed output of the coordinator is propagated to the distance. Thus the displayed lines for AAF are also a pessimistic approximation

of the distance between vehicle A and B. This effect is strongly decreased if the timespan of overlapping CC and DC requests is shorter (e.g., caused by a larger acceleration of vehicle B).

The green lines in Figure 8 illustrate the bounds of the minimum approximation using AADDs. Also for the distance, the effect of the uncertain mode switch from DC to CC ($t > 68s$) for the interval-based minimum approximation is not present when using the approximation based on AADDs.

The more precise approximation of the minimum operation using AADDs requires an increased simulation runtime, as compared to using the intervals only. The simulation of the presented driving scenario using AADD-based minimum approximation took about 4 minutes (on a standard PC), compared to 5 seconds using the interval-based approach.

6 Analyses

Our analyses are based on the additional information on deviations available when using Affine Arithmetic. In particular, we are interested in analyses of *deviation propagation*, which is possible through *Affine Arithmetic traces*. Such a trace is defined as a sequence of partial deviations.

First, let us define an auxiliary function $select$ operating on a given AAF, which extracts the partial deviation value with respect to an ϵ symbol:

$$select(\hat{x}(t), \epsilon_i) = \begin{cases} x_i, & i \in \mathcal{N}_{\hat{x}} \\ 0, & otherwise \end{cases} \tag{5}$$

Second, the trace sequence $\tau(\epsilon_i)$ is defined as

$$\tau(\epsilon_i) =< a_0, a_1, \ldots, a_k > \tag{6}$$

The length of the sequence k is limited by the defined simulation time or by the analysis setup. Values a_k represent partial deviations defined by the usage of $select$ functions associated with a specific uncertainty cause ϵ_i. For the application of the $select$ function, we have two possibilities. First, a trace for a certain point in time t, and second for a specific AAF $\hat{x}$ can be defined. These are the two proposed types of traces: temporal and structural.

A *temporal trace* includes partial deviation values from an AAF at increasing points in time. This is illustrated in Figure 9. Formally, a temporal trace is defined by the following usage of $select$ functions:

$$\begin{aligned} \tau_{temporal}(\epsilon_i) =& < a_0, a_1, \ldots, a_k > \\ a_0 =& \; select(\hat{x}(t_0), \epsilon_i) \\ a_1 =& \; select(\hat{x}(t_1), \epsilon_i) \\ & \cdots \\ a_k =& \; select(\hat{x}(t_k), \epsilon_i) \end{aligned} \tag{7}$$

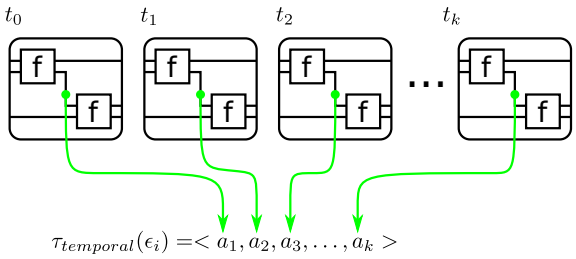

Figure 9: Temporal trace

A temporal trace facilitates a time-specific analysis of a single AAF value within the system. Strictly speaking, a conventional plot (see, e.g., Figure 6, etc.) of an AAF represents an accumulated representation of all included partial deviations and, hence, is a special case of a temporal trace. The tracing functionalities implemented in the presented AAF framework include more enhanced features for temporal tracing, e.g., assertion-driven trace start, selectable sampling frequency, limited trace length, etc. Results of an analysis process using temporal traces can be

- points in time where partial deviations exceed unspecified tolerance values, or have minimum or maximum values. These points in time can potentially indicate critical states of the system where the full behavior may violate system performance or safety requirements;
- detailed analysis of a deviation cause and its impact on the total radius of an AAF. This impact analysis may help designers to evaluate system states where a specific uncertainty cause has a dominant characteristic;
- the global impact of a partial deviation over the specified trace length. This is calculated by the integration of the values included in the trace;
- the temporal sensitivity of the system against the specified uncertainty cause. This can be evaluated by the first derivation of the trace represented by the absolute difference of sequence elements.

For the analysis of the ACC feature, we created two traces for the partial deviations included in the distance AAF, illustrated in Figure 10. The red line is the trace for the partial deviation associated with ϵ_1 (deviation of CC speed), the blue line for ϵ_2 (speed deviation of vehicle B). Based on these trace results, we can make the following qualitative statements for the analyzed ACC feature and the given scenario:

- The partial deviation value of the distance for ϵ_1 is negative due to the dependencies given by the physical model. Consider a positive value of ϵ_1, causing a

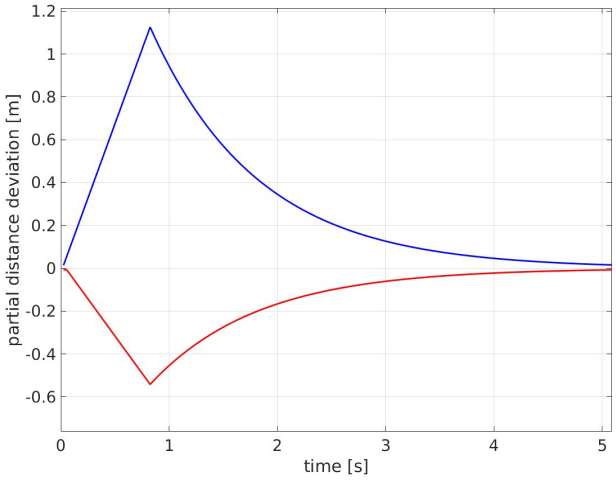

Figure 10: Temporal trace of the distance

speed value $v_A > v_{CC}$. This results in a faster approach towards vehicle B, i.e., a smaller distance, as compared to $v_A = v_{CC}$.

- The absolute impact of the deviation associated with ϵ_2 on the distance is higher than the one associated with ϵ_1, by a factor of 2 (equal to the ratio of the given speed deviations).

- The critical system state (in terms of distance deviation) occurs if and when the ACC feature is in CC mode. Due to the lack of any distance control function, deviations get continuously integrated, which also widens the total range (see the left part of Figure 10 until ACC switches to DC mode at $t = 0.825s$). This is also the case after the mode change from DC back to CC at $t = 73.3s$. The slope of the linear part indicates for the initial approach of vehicle A an increase of the deviation of the distance between the vehicles by $2m$ within one second.

- The first mode change from CC to DC at $t = 0.825s$ is picked as a critical point in time for our simulation and selected for analysis using a *structural* trace.

A *structural trace* is a path-selective analysis of the system. For its definition, a data-path represented as a set of AAFs through the system structure is specified. The *select* function is applied to each element of the set for a selected uncertainty cause ϵ_i and a specific point in time t (see Figure 11). Formally, the sequence elements of a structural

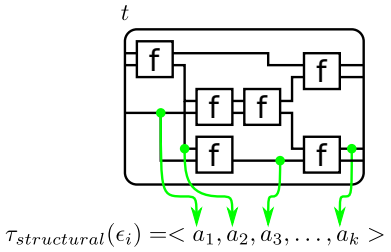

Figure 11: Structural trace

trace are defined as:

$$\begin{aligned} \tau_{structural}(\epsilon_i) =& < a_0, a_1, \ldots, a_k > \\ a_0 =& \; select(\hat{x_0}(t), \epsilon_i) \\ a_1 =& \; select(\hat{x_1}(t), \epsilon_i) \\ & \ldots \\ a_k =& \; select(\hat{x_k}(t), \epsilon_i) \end{aligned} \tag{8}$$

For the ACC example, we created two structural traces for ϵ_1 and ϵ_2, respectively, which we specify in one equation below. The point in time for recording the trace is set to $t_s = 0.825s$, where ACC switches from CC to DC mode in the given driving scenario. The traces are defined according to the following equation, and include all data values of the system ($\hat{v}_{CC}, \hat{v}_{DC}, \hat{v}_A, \hat{v}_B, \hat{d}$):

$$\begin{aligned} \tau_{structural1,2}(\epsilon_{1,2}) =< & \; select(\hat{v}_{CC}(t_s), \epsilon_{1,2}), \\ & select(\hat{v}_{DC}(t_s), \epsilon_{1,2}), select(\hat{v}_A(t_s), \epsilon_{1,2}), \\ & select(\hat{v}_B(t_s), \epsilon_{1,2}), select(\hat{d}(t_s), \epsilon_{1,2}) > \end{aligned}$$

Evaluating these formulas with the corresponding values results in the following numerical sequences:

$$\begin{aligned} \tau_{structural1}(\epsilon_1) =& < 0.694, 0.534, 0.649, 0, 0.534 > \\ \tau_{structural2}(\epsilon_2) =& < 0, 0.2, 0, 1.39, 1.11 > \end{aligned}$$

Defined traces may include partial deviation values from different physical quantities (speeds and a distance in this case). To facilitate the comparison of the partial deviation values included here, we divide them by the radius (computed by the rad operator) of the corresponding AAF (and multiply by 100) to get a percentage value.

$$\begin{aligned} & \tau_{structural1\%}(\epsilon_1) = \\ & \left\langle \frac{69.4}{rad(\hat{v}_{CC})}, \frac{53.4}{rad(\hat{v}_{DC})}, \frac{64.9}{rad(\hat{v}_A)}, \frac{0}{rad(\hat{v}_B)}, \frac{53.4}{rad(\hat{d})} \right\rangle = \\ & < 100\%, 72.75\%, 100\%, 0\%, 32.48\% > \\ & \tau_{structural2\%}(\epsilon_2) = \\ & \left\langle \frac{0}{rad(\hat{v}_{CC})}, \frac{20}{rad(\hat{v}_{DC})}, \frac{0}{rad(\hat{v}_A)}, \frac{13.9}{rad(\hat{v}_B)}, \frac{11.1}{rad(\hat{d})} \right\rangle = \\ & < 0\%, 27.25\%, 0\%, 100\%, 67.52\% > \end{aligned}$$

As a result, we can directly interpret these numbers for tracing the development of distance deviations in the system structure. ACC is in CC mode, hence the distance contribution of ϵ_1, 32.48%, results from $\hat{v}_{CC}$. The v_B deviation impact of 67.52% on the distance range is only propagated in the physical model, which is based on Newton's laws. Also the speed profile deviation of vehicle B is given by the simulated scenario and cannot, in principle, be improved by the ACC feature. Unfortunately, the major part of the integrating characteristic of the distance (partial deviation associated with ϵ_2) while ACC is in CC mode cannot be optimized. Thus, the only possibility to decrease the integration effect is to reduce the deviation of the CC speed parameter.

7 Conclusion

In this paper, we present a cyber-physical model of feature coordination using a semi-symbolic approach for simulation, based on Affine Arithmetic Forms. This approach can handle deviations and even systematic analyses of their propagation. We extended this approach for our model of Adaptive Cruise Control (viewed as a composite feature) by two new approximations of a minimum operator required for dealing with discrete discontinuities when switching between the Cruise Control and Distance Control features.

In contrast to more usual multi-run simulations, this approach can show the effects of deviations in a single simulation run. It also facilitates specific analyses of deviation propagation through temporal and structural traces. These revealed for our specific cyber-physical model, e.g., that the deviations caused from uncertainty of the speed by Cruise Control are propagated towards more and more uncertainty on the distance between vehicles, while the uncertainties from the speeds of both vehicles are reduced by Distance Control. Hence, this approach can help to gain a better understanding of different deviation propagations within such a semi-symbolic model.

Acknowledgment

The FeatureOpt project (No. 849928), is funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" between June 2015 and May 2018. More information can be found at `https://iktderzukunft.at/en/`.

References

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, Cambridge, MA, USA, 2008.

[2] J. Comba and J. Stolfi. Affine atithmetic and its applications to computer graphics. In *Anais do VI Simposio Brasileiro de Computacao Grafica e Processamento de Imagens (SIBGRAPI)*, 1993.

[3] D. Grabowski, M. Olbrich, C. Grimm, and E. Barke. Range arithmetics to speed up reachability analysis of analog systems. In *FDL'07*, pages 38–43, 2007.

[4] C. Grimm, W. Heupke, and K. Waldschmidt. Refinment of Mixed-Signal Systems with Affine Arithmetic. In *DATE*, 2004.

[5] M. Jackson and P. Zave. Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering (TSE)*, 24(10):831–847, 1998.

[6] C. Luckeneder, M. Rathmair, and H. Kaindl. Investigating and coordinating safety-critical feature interactions in automotive systems using simulation. In *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS-50)*. CC-BY-NC-ND, 2017.

[7] C. Radojicic, C. Grimm, A. Jantsch, and M. Rathmair. Towards Verification of Uncertain Cyber-Physical Systems. In *Proceedings 3rd International Workshop on Symbolic and Numerical Methods for Reachability Analysis (SNR 2017)*, pages 1–17. EPTCS 247, April 2017.

[8] M. Rathmair, C. Luckeneder, and H. Kaindl. Minimalist qualitative models for model checking cyber-physical feature coordination. In *Proceedings of the 23rd Asia-Pacific Software Engineering Conference (APSEC)*, USA, Dec 2016. IEEE.

[9] M. Rathmair, F. Schupfer, C. Radojicic, and C. Grimm. Extended framework for system simulation with affine arithmetic. In *Proceeding of the 2012 Forum on Specification and Design Languages*, pages 168–175, Sept 2012.

[10] J. Stolfi and L. de Figueired. An introduction to affine arithmetic. In *TEMA Trend. Mat. Apl. Comput.*, volume 4, pages 297–312, 2003.

[11] H. Winner and M. Schopper. Adaptive cruise control. In *Handbuch Fahrerassistenzsysteme*, pages 851–891. Springer, 2015.